AMENDMENTS TO THE SPECIFICATION:

Page 1, 1st paragraph (inserted via Preliminary Amendment filed March 29, 2006):

This application is the US national phase of international application PCT/IB2004/003334, filed 30 September 2004, which designated the U.S. and claims priority of EP 03292414.4, filed 30 September 2003 (published as EP 1673697 (A2) as of June 28, 2006), the contents of each of which are hereby incorporated by reference.

Page 1, top of page: delete "**OPERATING SYSTEMS**" and insert the following heading:

TECHNICAL FIELD

Page 1, between 1st and 2nd paragraphs, insert the following heading:

BACKGROUND

Pages 1-2, bridging paragraph:

For such tasks, therefore, real time operating systems have been developed; one example is ChorusOS (also known as Chorus) and its derivatives. Chorus is available as open source software. ~~from: http://www.experimentalstuff.com/Technologies/ChorusOS/index.html and Jaluna at http://www.jaluna.com/~~

Page 2, 1st full paragraph:

It is described in "ChorusOS Features and Architecture overview," Francois Armand, Sun Technical Report, August 2001, 222p. ~~, available from: http://www.jaluna.com/developer/papers/COSDESPERF.pdf~~

Page 3, 2nd - 4th full paragraphs:

Another approach is that of ADEOS (Adaptive Domain Environment for Operating Systems), described in a White Paper, at ~~http://opersys.com/ftp/pub/Adeos/adeos.pdf~~

ADEOS provides a nanokernel which is intended, amongst other things, for running multiple operating systems although it appears only to have been implemented with Linux. One proposed use of ADEOS was to allow ADEOS to distribute interrupts to RTAI (Realtime Application Interface for Linux) ~~for which see:~~

~~http://www.aero.polimi.it/~rtai/applications/~~.

Page 3, between 5th and 6th full paragraphs, insert the following heading:

BRIEF SUMMARY

Page 3, 6th full paragraph:

An object of the present invention is to provide an improved system, method and computer program for running multiple operating systems simultaneously, even when the systems are designed for different purposes. In particular, the present invention aims to allow one of the operating systems (for example, a real time operating system[[s]]) to perform without disturbance, and the other (for example, a general purpose operating system) to perform as well as possible using the remaining resources of the computer.

Page 4, final paragraph:

Handling of such interrupts is thus deferred until no critical task in the primary operating system is occurring. When they are eventually actioned, however, the routines of the secondary operating system may operate in substantially unmodified

fashion so that the behaviour is (except for the delay) as expected by the secondary operating system.

Page 5, between 1st and 2nd paragraphs, insert the following heading:

BRIEF DESCRIPTION OF THE DRAWINGS

Page 5, 7th paragraph:

All are available free of charge from Intel Corporation, PO Box 7641, Mt. Prospect, IL 60056-7641. ~~and can be downloaded from~~ http://www.intel.com

Page 7, 2nd paragraph:

FIG. 4 shows the components of a hardware resource dispatcher forming part of FIG. 2b;

Page 7, 15th paragraph:

Figure 16 shows typical states of a TSS stack[[.]];

Page 8, line 1: delete "**Introduction**" and insert the following heading:

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Page 8, 1st - 2nd paragraphs:

System Hardware

A computer system to which the system is applicable 100 comprises a central processing unit (CPU) 102, such as a Pentium 4™ CPU available from Intel Corporation, or PowerPC CPU available from Motorola (the embodiment has been implemented on both), coupled via a system bus 104 (comprising control, data and address buses) to a read-only memory (ROM) chip 106; one or more banks of random

access memory (RAM) chips (108); disk controller devices 110 (for example, Integrated Device Electronics (IDE) or Small Computer System Interface (SCSI) controllers, connected to a floppy disk drive, a hard disk drive, and additional removable media drives such as Digital Versatile Disc (DVD) drives); one or more input/output ports (112) (for example, one or more Universal Serial Bus (USB) port controllers, and/or parallel port controllers for connection to printer and so on); an expansion bus 114 for bus connection to external or internal peripheral devices (for example, the Peripheral Component Interconnect (PCI) bus); and other system chips 116 (for example, graphics and sound devices). Examples [[or]] of computers of this type are personal computers (PCs) and workstations. However, the application of the invention to other computing devices such as mainframes, embedded microcomputers in control systems, and Personal Digital Assistants (PDAs) (in which case some of the indicated devices such as disk drive controllers may be absent) is also disclosed herein.

Management of Software

Referring to FIG. 2a, in use, the computer 100 of FIG. 1 runs resident programs comprising operating system kernel 202 (which provides the output routines allowing access by the CPU to the other devices shown in FIG. 1); an operating system user interface or presentation layer 204 (such as X Windows); a middleware layer 206 (providing networking software and protocols such as, for instance, a Transmission Control Protocol/Internet Protocol (TCP/IP) stack) and applications 208a, 208b, which run by making calls to the Application Programming Interface (API) routines forming the operating system kernel 202.

Page 8, 8th full paragraph:

The operating systems are not treated equally by the embodiment. Instead, one of the operating systems is selected as the "critical" operating system[[s]] (this will be the real time operating system), and the or each other operating system is treated as a

"non-critical" or "secondary" operating system[[(s)] (this will be the or each general purpose operating system such as Linux).

Page 10-11, bridging paragraph:

For example, a parallel printer port might be statically allocated to the general purpose operating system 202, which will often run applications which will need to produce printer output. On the other hand, an Integrated Services Digital Network (ISDN) digital line adapter port may be permanently allocated to the real time operating system 201 for communications. This static allocation of devices wherever possible means that each operating system can use its existing drivers to access statically allocated devices without needing to call the hardware resource dispatcher. Thus, there is no loss in execution speed in accessing such devices (as there would be if it acted as a virtual machine or emulator).

Page 12, 2nd paragraph:

In this embodiment, the computer 100 is an Intel 386 family processor (e.g. a Pentium processor) (step 302). The critical operating system 201 was the C5 operating system (the real time microkernel of Jaluna-1, an open-source version of the fifth generation of the ChorusOS system, available for open source~~, free download from http://www.jaluna.com~~).

Page 12, 3rd paragraph:

In step 306, the ChorusOS operating system kernel 201 is modified for operating in multiple operating system mode, which is treated in the same way [[s]] by porting to a new platform (i.e. writing a new Board Support Package to allow execution on a new computer with the same CPU but different system devices). The booting and initialisation sequences are modified to allow the real time operating system to be

started by the hardware resource dispatcher, in its allocated memory space, rather than starting itself. The hardware-probing stage of the initialisation sequence is modified, to prevent the critical operating system from accessing the hardware resources which are assigned to other secondary systems. It reads the static hardware allocation table from the hardware resource dispatcher to detect the devices available to it.

Page 13, 1st paragraph:

Trap calls [[2012]] are added to the critical operating system, to detect states and request some actions in response. A trap call here means a call which causes the processor to save the current context (e.g. state of registers) and load a new context. Thus, where virtual memory addressing is used, the address pointers are changed. For example, when the real time operating system 201 reaches an end point (and ceases to require processor resources) control can be passed back to the hardware resource dispatcher, issuing the "idle" trap call, to start the secondary operating system. Many processors have a "halt" instruction. In some cases, only supervisor-level code (e.g. operating systems, not applications) can include such a "halt" instruction. In this embodiment, all the operating systems are rewritten to remove "halt" instructions and replace them with an "idle" routine (e.g. an execution thread) which, when called, issues the "idle" trap call.

Page 13, 2nd full paragraph:

Additional "virtual" drivers [[2014]] are added which, to the operating system, appear to provide access to an input/output (I/O) bus, allowing data to be written to the bus. In fact, the virtual bus driver [[2014]] uses memory as a communications medium; it exports some private memory (for input data) and imports memory exported by other systems (for output data). In this way, the operating system 201 (or an application running on the operating system) can pass data to another operating system (or

application running on it) as if they were two operating systems running on separate machines connected by a real I/O bus.

Pages 13-14, bridging paragraph:

In step 310, the secondary operating system kernel 202 is modified to allow it to function in a multiple operating system environment, which is treated as [[a]] new hardware architecture. As in step 306, the boot and initialisation sequences are modified, to allow the secondary operating system to be started by the hardware resource dispatcher, and to prevent it from accessing the hardware resources assigned to the other systems, as specified in the hardware resource dispatcher table. As in step 306, trap calls [[2022]] are added, to pass control to the hardware resource dispatcher.

Page 14, 1st full paragraph:

Native drivers for shared system devices are replaced by new drivers [[2028]] dealing with devices which have been virtualized by the hardware resource dispatcher (interrupt controller, I/O bus bridges, the system timer and the real time clock). These drivers execute a call to virtual device handlers 416 of the hardware resource dispatcher in order to perform some operations on a respective device of the computer 100. Each such virtual device handler 416 of the hardware resource dispatcher is paired with a "peer" driver routine in the critical operating system, which is arranged to directly interact with the system device. Thus, a call to a virtual device handler is relayed up to a peer driver in the critical system for that virtualized device, in order to make real device access. As in step 306, read and write drivers [[2024]] for the virtual I/O bus are provided, to allow inter-operating system communications.

Pages 14-15, bridging paragraph:

The interrupt service routines of the secondary operating system are modified, to provide virtual interrupt service routines [[2026]] each of which responds to a respective virtual interrupt (in the form of a call issued by an interrupt handler routine 412 of the hardware resource dispatcher), and not to respond to real interrupts or events. Routines of the secondary operating system (including interrupt service routines) are also modified to remove masking of hardware interrupts (at least in all except critical operations). In that way, the secondary operating systems 202, . . . are therefore pre-emptable by the critical operating system 201; in other words, the secondary operating system response to a virtual interrupt can itself be interrupted by a real interrupt for the critical operating system 201. This typically includes: masking/unmasking events (interrupts at processor level); saving/restoring events mask status; identifying the interrupt source (interrupt controller devices); masking/unmasking interrupts at source level (interrupt controller devices).

Page 15, 1st - 3rd paragraphs:

New virtual device drivers [[2028]] are added, for accessing the shared hardware devices (the 1/0 bus bridges, the system console, the system timer and the real time clock). These drivers execute a call to virtual device handlers 416 of the hardware resource dispatcher in order to write data to, or read data from, a respective device of the computer 100.

To effect this, the Linux kernel [[207]] is modified in this embodiment by adding new virtual hardware resource dispatcher architecture sub trees (nk-i386 and nk-ppc for the I-386 and PowerPC variants) with a small number of modified files. Unchanged files are reused in their existing form. The original sub-trees are retained, but not used.

In step 312, the hardware resource dispatcher 400 is written. The hardware resource dispatcher comprises code which provides routines for the following functions [[as]] (as shown in FIG. 4):

- booting and initialising itself (402);
- storing a table [[(403)]] which stores a list of hardware resources (devices such as ports) and an allocation entry indicating to which operating system each resource is uniquely assigned;
- booting and initialising the critical operating system that completes the hardware resource dispatcher allocation tables (404);
- booting and initialising secondary operating systems (406);
- switching between operating systems (408);
- scheduling between operating systems (410);
- handling interrupts (using the real time operating system interrupt service routines, and supplying data where necessary to the virtual interrupt service routines of the secondary operating systems) (412);
- handling trap calls from each of the operating systems (414);
- handling access to shared devices from the secondary operating systems (416);
- handling inter-operating system communications on the virtual I/O bus (418).

Page 17, 2nd paragraph:

The hardware resource dispatcher is arranged to provide mechanisms to handle processor exceptions (e.g. CPU interrupts or co-processor interrupts) as follows:

- firstly, to intercept processor exceptions through the critical operating system;

- secondly, to post a corresponding virtual exception to one or more secondary operating systems; to store that data and, when the scheduler next calls that secondary operating system, to call the corresponding virtual interrupt service routine [[2026]] in the secondary operating system;
- thirdly, to mask or unmask any pending virtual exceptions from within secondary operating systems.

Pages 20, 1st full paragraph:

This system image is stored in persistent storage (e.g. read only memory for a typical real time device such as a mobile telephone or Private Branch Exchange (PBX)). The remaining banks of memory are available to be allocated to each operating system as its environment, within which it can load and run applications.

Page 22, 5th and 6th full paragraphs:

"Fast Error Recovery in CHORUS/OS. The Hot-Restart Technology," [[.]] Abrossimov, F. Hermann., J. C. Hugly, et al., Chorus Systems Inc., Technical Report, August 1996, 14 p. ~~available from:~~

~~http://www.jaluna.com/developer/papers/CSI-TR-96-34.pdf~~

Page 23, 2nd full paragraph:

At some point, an interrupt or event will occur. For example, a packet may be received at a data port, causing an interrupt to allow it to be processed by the real time operating system executing the UDP/IP stack. Alternatively, a user may manipulate a keyboard or mouse, causing an interrupt to operate the Graphical User Interface (GUI) of the second operating system 202 for interaction with the word processing application 208. Alternatively, the system clock may indicate that a predetermined time has

elapsed, and that an application should commence re-execution, or an operating system function should execute.

Page 26, 5th full paragraph:

Referring to FIG. 9b, the changes necessary to migrate this arrangement to one in which the first and second operating systems run on different computers 100[[, 101]] are small; it is merely necessary to change the drivers used by the operating systems, so that they use drivers for a real bus 103 rather than the virtual bus drivers. The system is therefore made more independent of the hardware on which it operates.

Page 28, 1st - 2nd full paragraphs:

Some other aspects of the overall (host/target) debugging architecture according to this embodiment are similar to those for the Chorus and C5 debugging systems, described in the document "C5 1.0 Debugging Guide" published by Jaluna, ~~and available at:~~

~~http://www.jaluna.com/doc/c5/html/DebugGuide/book1.html~~